Remarks

This responds to the Office action mailed March 30, 2005 ["Action"]. Reconsideration of the application is respectfully requested in view of the following remarks. Claims 10-14 and 41-45 are pending in the application. No claims have been allowed. Claims 10 and 41 are independent.

Moot Rejections

The Action contains two rejections which the Applicants respectfully traverse as moot. The first is a double patenting rejection of claims 1-9, 15, 24-40, and 51-56 over various claims of U.S. Patent No. 6,415,334. The Action also rejects claims 16-23 and 46-50 under 35 U.S.C. 102(e) as being anticipated by U.S. Patent No. 6,085,030.

However, claims 1-9, 15-40, and 46-50 were previously cancelled by a preliminary amendment filed concurrently with the Application on December 26, 2001. Because of this, Applicants believe the rejections to be moot and will not belabor traversing each rejection independently.

Rejections Under 35 U.S.C. § 102(a)

The Action also rejects claims 10-14 and 41-45 under 35 U.S.C. § 102(a) as being anticipated by "Give ActiveX-based Web Pages a Boost with the Apartment Threading Model" by Platt ["Platt"]. Applicants respectfully submit the claims in their present form are allowable over the cited art. For a 102(a) rejection to be proper, the cited art must show each and every element as set forth in a claim. (*See* MPEP § 2131.) However, the cited art does not describe each and every element. Accordingly, applicants request that all rejections be withdrawn.

Claim 10

Claim 10 recites, in part:

> *requesting the instantiating thread to marshal a second reference to the object* to the current thread;
>
> when the second reference is received,
>
> *determining whether the second reference is the same as the first reference;*

when the references are the same, invoking the method of the object from the current thread; and

when the references are not the same, requesting that the instantiating thread invoke the method of the object.

For example, the passage at page 14, line 21 to page 15, line 8 of the Application describes:

> The standard marshaling technique uses a proxy object and a stub object. Each time an object is marshaled using standard marshaling a new proxy object is instantiated. Thus, marshaled pointers to the same COM object will have different values, since they point to different proxy objects. However, pointers to the same COM object that are marshaled using free-threaded marshaling will have the same value, since they point to the COM object itself and not to a proxy object. When the thread system attempts to first invoke a method of a returned COM object, the thread system requests the returning thread to marshal a pointer to the COM object to the current thread. The returning thread marshals the pointer according to the marshaling technique of the COM object. When the thread system receives the marshaled pointer, the thread system compares that pointer to the pointer to the IUnknown interface for that returned COM object stored in the wrapper object. If both pointers to the IUnknown interface are the same, then the COM object is thread-safe and the thread system can directly invoke the methods of the COM object from any thread. However, if the pointers are not the same, then standard marshaling created a new proxy object to which the marshaled pointer points and thus the COM object is not thread-safe.

As a further example, at FIG 9. and page 15, line 27 to page 16, line 4 of the Application describe:

> In step 903, the Java VM waits for completion of the marshaling. In step 904, if the marshaled pointer to the IUnknown interface is the same as the pointer to the IUnknown interface in the wrapper object, then the Java VM continues at step 905, else the Java VM continues at step 906. In step 905, the Java VM sets the home thread field to null because the COM object is identified as being thread-safe. In step 906, the Java VM sets the tentative home thread flag to indicate not tentative and continues with its processing.

As the examples describe, in one instantiation of the invention, the current thread does not know for sure whether or not it has a direct reference to a thread-safe object or a reference through a proxy to a single-threaded object. In order to determine what type the object is, it is useful to compare this "first" reference to a known direct reference. Thus, the current thread requests that the instantiating thread provide a "second" reference, since it can trust the instantiating thread to be able to directly reference the object. By then comparing these references, the current thread can find out if the reference it has is direct, which allows it to

directly invoke methods, or if it has a reference to a proxy, in which case it requests method invocation through the instantiating thread.

Claim 10 stands rejected over Platt. However, Platt does not describe each and every element of the claim as required by § 102. The Action alleges that Platt describes various aspects of claim 1 at page 15, paragraph 2, and page 13, paragraph 2

Platt's description of "marshalling" on page 13 does not describe "requesting the instantiating thread to marshal a second reference to the object" as recited by claim 1. On page 13, paragraph 2, Platt describes:

> Marshalling *an interface to another thread* is accomplished via two API functions.... The destination thread calls the API function CoGetInterfaceAndReleaseStream, passing a pointer to the stream containing the marshalling information. *This function returns a pointer to the original interface* that the new thread may now call. *This is a proxy that actually marshals the call into the original thread rather than a direct connection.*

Thus, while Platt describes marshalling some sort of reference, the cited description of Platt explicitly describes the reference as being to "an interface" and "a proxy" that is different than "a direct connection." This does not show "requesting the instantiating thread to marshal a second reference to the object" as recited in claim 10.

Platt's cannot show "determining whether the second reference is the same as the first reference" as recited in claim 1, as it requires that COM determine whether objects are free-threaded or single-threaded based on registry information. On page 8, paragraphs 2 and 3, Platt describes:

> A control that supports the single-threaded apartment model signals this to COM by making an entry in the registry as shown in Figure 6....
>
> If you are writing a control with MFC version 4.2 or later and do not want to support apartment-model threading, *you must change this parameter to zero.* Conversely, if you are upgrading a control that was originally generated with MFC version 4.1 or earlier and now want to support the apartment model, *this parameter as generated in your original code will be FALSE and you must change it....*

Thus, as described in Platt, registry entries must be changed in order to identify whether a particular object supports apartment-model threading or is a free-threaded object. Furthermore, on page 15, paragraph 2, Platt describes leaving determinations of the type of threading an object uses up to COM:

> How do I know if the interface really needed to be marshaled at all? It might have been a free-threaded interface that wouldn't care if it was called from another thread. *I don't know, but fortunately I don't have to know or care.* If I simply make the function calls to marshal the interface as shown above, and the interface actually belongs to an object that doesn't require marshalling, *COM will simply not set up the marshalling code* .... I follow a few simple rules and COM intercedes to the extent necessary. I don't have to know what that extent is in order to use it, and I'd just as soon not have to think about it.

As this passage makes clear, Platt not only does not show "determining whether the second reference is the same as the first" but rather teaches a deliberate indifference to this determination. Thus, it cannot show the recited language of claim 10.

For at least these reasons, claim 10, and its dependent claims 11-14 are allowable at this time.

Claim 41

Claim 41 recites, in part:

> *requesting the instantiating thread to marshal a second reference to the object* to the current thread;
> when the second reference is received,
> *determining whether the second reference is the same as the first reference;*
> when the references are the same, invoking the method of the object from the current thread; and
> when the references are not the same, requesting that the instantiating thread invoke the method of the object.

Applicants note the similarity between the language of claim 41 and the language of claim 10. Thus, for at least the reasons described above with reference to claim 10, claim 41, and it's dependent claims 42-45, are allowable at this time.

Request For Interview

If any issues remain, the Examiner is formally requested to contact the undersigned attorney prior to issuance of the next Office Action in order to arrange a telephonic interview. It is believed that a brief discussion of the merits of the present application may expedite prosecution. Applicants submit the foregoing formal Amendment so that the Examiner may fully evaluate Applicants' position, thereby enabling the interview to be more focused.

This request is being submitted under MPEP § 713.01, which indicates that an interview may be arranged in advance by a written request.

Conclusion

The claims in their present form should now be allowable. Such action is respectfully requested.

Respectfully submitted,

KLARQUIST SPARKMAN, LLP

By _____
Stephen A. Wight
Registration No. 37,759

One World Trade Center, Suite 1600
121 S.W. Salmon Street
Portland, Oregon 97204
Telephone: (503) 226-7391
Facsimile: (503) 228-9446